Borland AppServer

An Integrated Solution for Developing, Deploying, and Managing Distributed Multi-tier Applications.

August 1998

Borland

Contents

Borland is helping to create the foundation on which tomorrow's mission-critical applications will be built—an open, distributed, object-based architecture for the new global enterprise. The leading supplier of distributed object computing and enterprise application tools, Borland develops pioneering products that enable IT organizations to protect their investments in existing applications while moving to distributed object computing and embracing the new opportunities presented by the Internet.

This white paper discusses why enterprises are turning to multi-tier, distributed architectures, and describes problems they are encountering as they make this shift. The paper also explains why an application server is essential for multi-tier, distributed applications, and provides an overview of existing application server technology. After explaining why current application server solutions are not meeting all of the needs of enterprises, this paper introduces Borland AppServer—the revolutionary new integrated architecture for developing, deploying, and managing distributed multi-tier applications.

Introduction

As distributed object applications become mainstream, and Web-based business takes hold, enterprises are shifting to the middle tier. By placing business logic on middle-tier servers, enterprises can easily update applications for thousands of clients—without requiring clients to perform installations.

However, middle-tier architectures are much more difficult to develop than traditional client/server applications, and pose a set of unique challenges for deployment and management.

For enterprises to make the leap to multi-tier, distributed architectures, mainstream IT developers must be able to develop sophisticated distributed applications using a visual environment. IT must be able to easily deploy these applications in a test and production environment that is shared with administrators. And administrators need to manage these de-centralized applications from a central location.

What enterprises need is a solution that integrates the key technologies necessary to simplify the development, deployment, and management of distributed, multi-tier applications. The solution is an application server. There are currently two types of application servers on the market: Web application servers and legacy application servers.

Web application servers enable easy development of Internet applications, and ***Legacy application servers*** rely on existing systems such as TP Monitors. These types of application servers are fully described later in the paper. However, neither of these application servers fully meets enterprise requirements for the Web era.

Borland delivers the next generation ***Enterprise application server***—an integrated solution that combines the best of application server technology to meet the needs of enterprises as they shift to multi-tier, distributed, Web-based solutions.

Enterprises Shift to the Middle-tier

Enterprises are shifting to middle-tier servers in droves as distributed applications become mainstream. Distributed object applications offer enterprises improved fault tolerance and scalability, faster time-to-market by reusing application components, and a way to tie disparate heterogeneous environments together using the power of the Web.

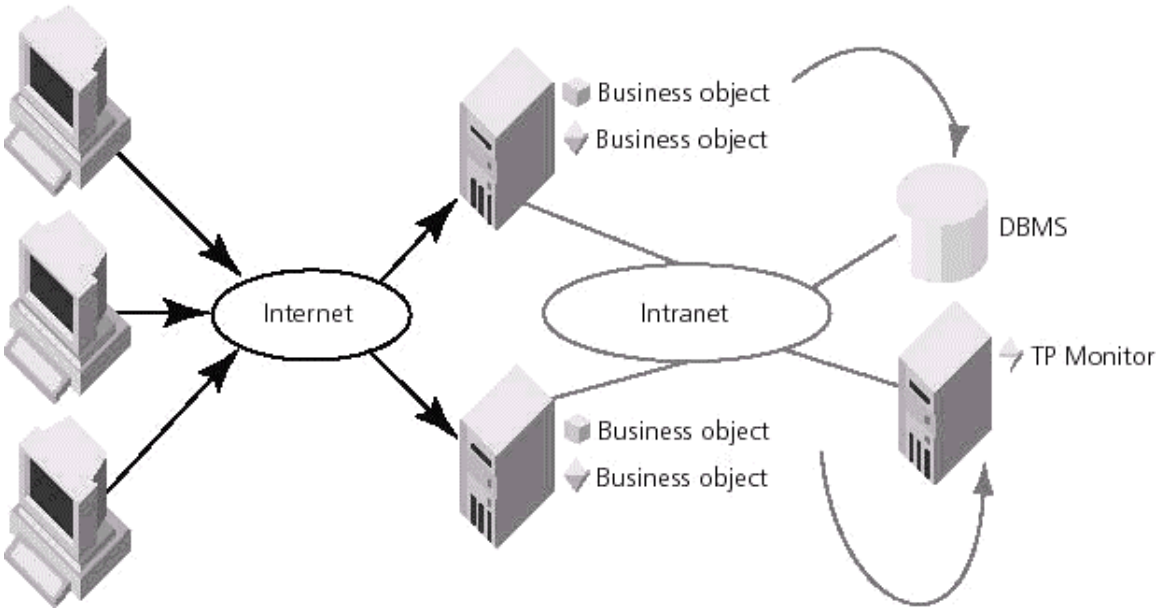

Figure 1. Middle-tier architectures used by enterprises.

By partitioning applications into components, and shifting business logic onto middle-tier servers, IT can vastly reduce turnaround time for development of applications by reusing the logic in these middle-tier server objects. Another key benefit of middle-tier architectures is the ability to locate business logic on the server rather than the client. This means that if code needs to change, it can be changed in one place rather than in each of 10,000 client applications.

Multi-tier distributed object architectures also offer significant advantages for enterprises as they seek to integrate new Web-based business with existing systems. Web-based business applications require a thin-client architecture to support browser-based clients. These clients need to interact with intranet-based resources, but often have limited system resources that make it difficult to download applets.

Using server-side, middle-tier solutions, IT can alleviate the burden from these clients by transferring the bulk of business logic to middle tier servers-leaving clients with a thin interface to the rich functionality users crave.

These server-side solutions also bridge heterogeneous platforms and integrate with legacy systems, enabling Web-based applications to integrate with existing enterprise systems.

Challenges Posed by Multi-tier, Distributed Applications

Although multi-tier, distributed architectures offer numerous advantages to enterprises, the shift to this new architecture comes with its own unique growing pains:

- Developing a multi-tier, distributed application is complex
- Deploying the hundreds of components that make up a distributed application is challenging
- Managing thousands of distributed components is daunting

Developing a Multi-tier, Distributed Application Is Complex

Highly technical, advanced programmers have been developing multi-tier, distributed applications for some time. But these developers have deep knowledge of underlying system-level complexities such as concurrency, locking, transaction management, security, and scalability. They also understand how to manage access to system resources such as threads, memory, database connections, and network connections.

But the complexity of these issues has limited the spread of multi-tier development because mainstream IT developers typically have more experience with business logic development rather than system-level programming.

Deploying the Hundreds of Components that Make Up a Distributed Application Is Challenging

Most distributed applications are comprised of hundreds of components. Each component has properties that must be configured to determine how it starts up, if it logs information, and so on. Many times, the way these properties are set depends on the platform where the component resides.

Furthermore, the test and production environments for these distributed applications are typically disconnected, preventing developers from making intelligent modifications to these

applications to improve performance and scalability in real-time deployments. Once these applications are deployed, keeping track of thousands of decentralized objects is challenging.

Managing Thousands of Distributed Components Is Daunting

Once distributed applications are deployed, administrators need to ensure that they keep running. Components of distributed applications can reside anywhere across the enterprise, and experience failures from all manner of causes including system failure, network interruptions, and application errors.

Administrators must quickly discover that a component has failed, and then take immediate steps to restart the component, or to start a replica of the component and redirect client traffic.

However network and system management, and particularly SNMP, all approach management from the point of view of a host. Hosts perform particular functions, have software running on them, support SNMP MIBs, and so on. But a distributed application does not run on a host—it runs on an interconnected network of hosts. By looking at a single host, it is impossible to tell whether an application is running.

To administer a distributed application, administrators must administer the entire network—this is a daunting task if you do not have the right tools.

Requirements for Multi-tier, Distributed Applications

To make multi-tier, distributed architectures viable for the enterprise, the following requirements must be met:

- **Easy development by mainstream IT.** Multi-tier, distributed architectures have required too much system-level knowledge of diverse technologies (such as databases, TP Monitors, security, and CORBA) to make them desirable for mainstream IT. For these architectures to be practicable for enterprises, mainstream IT developers must be able to quickly and easily build sophisticated, multi-tier solutions without understanding underlying system-level complexities. They must be able to develop these applications—and enable transactions and security—using a visual Integrated Development Environment (IDE). Enterprise JavaBeans (EJB) should also be supported for easy development of server-side Java components.

- **Simplified deployment with integrated test and production environments.** IT must be able to easily test these distributed solutions, make modifications to improve performance, and then work with administrators in the same environment to deploy them for production usage. To simplify deployment, IT needs a solution that provides a centralized view of all components that reside on hosts on the network, and an easy way to set properties for the components of a distributed application.

- **Centralized management.** IT must be able to easily manage hundreds of distributed applications, many of which have thousands of components that are distributed across the enterprise. To do so, they need a tool that provides centralized management and control over distributed, decentralized applications, along with automatic problem detection and correction capabilities.

- **Robust framework for business-critical, enterprise applications.** For distributed, multi-tier applications to be useful to the enterprise, they must be based on a framework that provides transaction handling, security, failover, load-balancing, scalability, and high performance.

- **Open and standards-based.** Enterprises need solutions that are open and standards-based so that they can be interoperable with other software that is built on standards. For example, distributed applications must integrate with back-end systems applications like SAP and Peoplesoft.

- **Integrated with databases and legacy systems.** Distributed applications must access corporate data that resides in popular databases (such as Oracle and Sybase), or is accessed via TP Monitors (such as CICS and Tuxedo).

- **Support for diverse environments.** Enterprises have diverse environments that must be supported to enable multi-tier, distributed applications. On the server side, platforms such as Windows NT and UNIX must be supported for server-based components. A variety of clients must be able to access these server-based components, including HTML, Java applets, Java applications, dynamic HTML, and C++ applications.

An Enterprise Application Server Is the Answer

Enterprises that are moving to multi-tier, distributed environments need an application server. An application server ties together disparate technologies to make development, deployment, and management of multi-tier, distributed applications easier.

Currently, there are a plethora of "application server" technologies on the market. These technologies break down into two basic categories:

- Web application server
- Legacy application server

Web Application Server

The Web application server provides a development environment for Web-based HTML applications. It enables HTML authoring of Web-based client/server applications. In this architecture, a Web application server resides on the Web server, and handles incoming client requests. ODBC and JDBC are used to connect with databases.

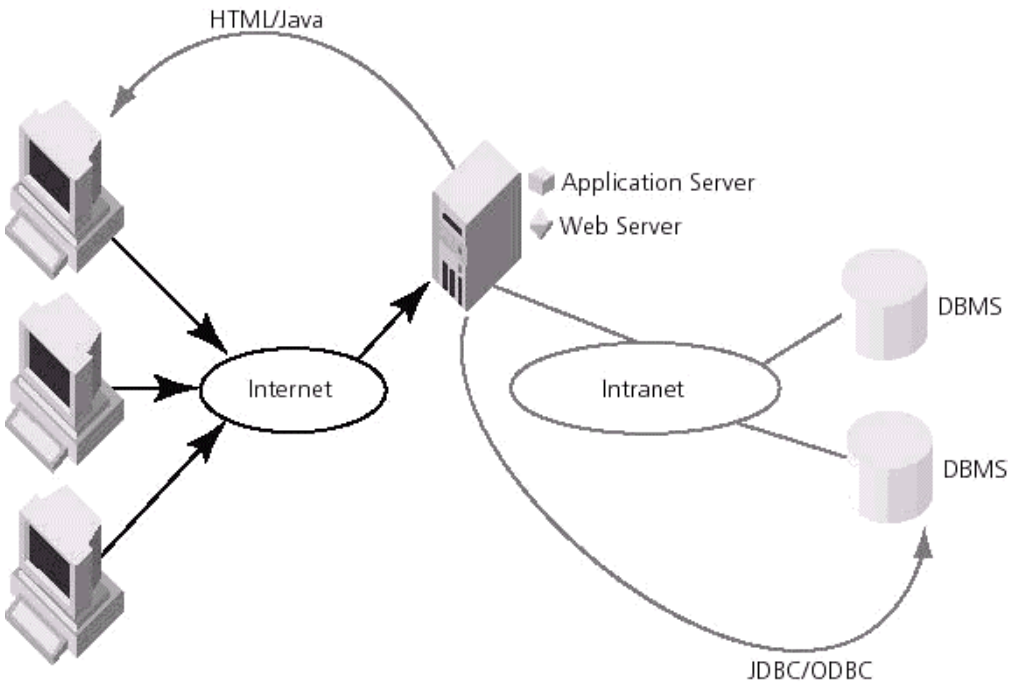

Figure 2. Architecture of Web application servers.

This type of application server is typically easy to use and Java-oriented with support for Enterprise JavaBeans for server-side component development.

However, these solutions do not meet the requirements of mission-critical enterprise applications-they do not provide support for transactions, they provide little security, they do not access legacy systems, and they typically have less than optimal performance.

Advantages	Drawbacks
Built for Internet applications	No support for transactions
Integrated with the Web server	Low performance
Easy development	No systems management
Load balancing for scalability	Little support for existing systems

Examples of a Web application server are solutions provided by Netscape, NetDynamics, and WebLogic.

Legacy Application Server

The legacy application server provides a business-critical application infrastructure with transactions handling, security, failover, and load balancing to integrate data in existing, legacy systems such as TP Monitors. However, many of these solutions bootstrap distributed objects into a client/server architecture, and thereby provide cobbled support for Web-based activities.

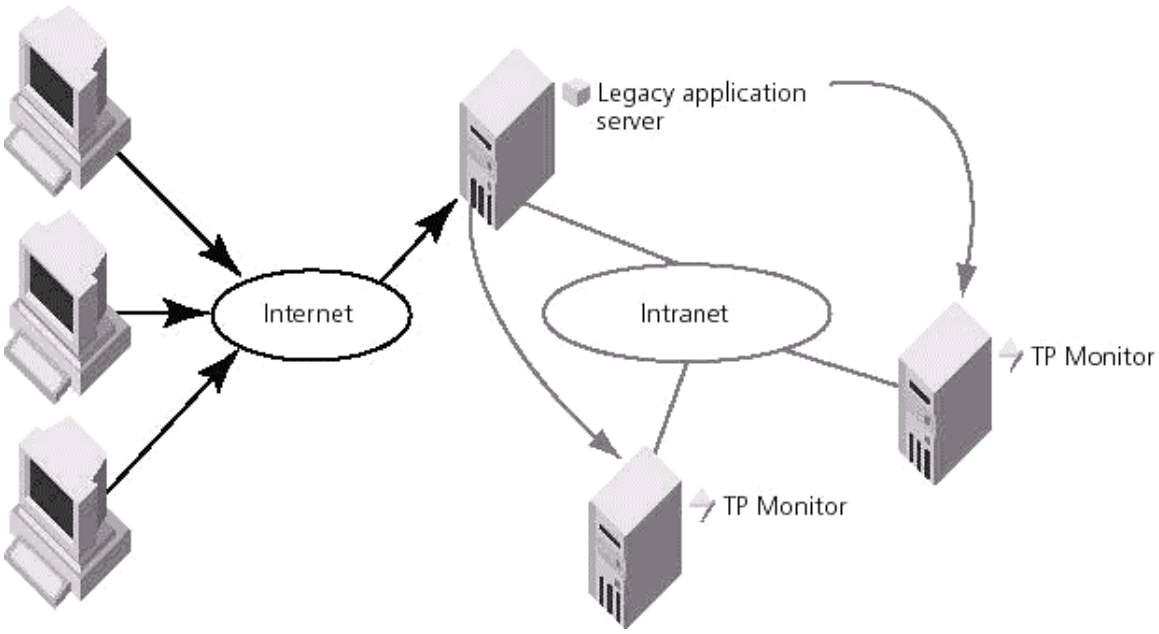

Figure 3. Architecture of legacy application servers.

Advantages	Drawbacks
Proven enterprise, mission-critical application reputation	Not built for a distributed, Web environment
Transactions handling for intranet-based applications	Low performance for Internet transactions
Security for intranet-based enterprise applications	Complex development environment
	Limited language support
	No integration with the Web server

Examples of a legacy application server are solutions provided by BEA M3 and IBM Component Broker.

Enterprise Application Server

Most existing application servers do not provide a complete solution. What enterprises need is the best of both worlds. They need an application server built on an enterprise-class infrastructure that provides transactions, security, and centralized management for Web-based distributed applications. Enterprises need Borland AppServer.

Borland AppServer—the Integrated Environment for Multi-tier, Web-based Applications

Borland has recognized the need for a solution that simplifies the development, deployment, and management of distributed applications for the Web. To meet this need, Borland is providing the Borland AppServer—an integrated, end-to-end solution that offers GUI-based development, deployment, and management on a foundation of enterprise-strength, standards-based transactions, security, and object communications software.

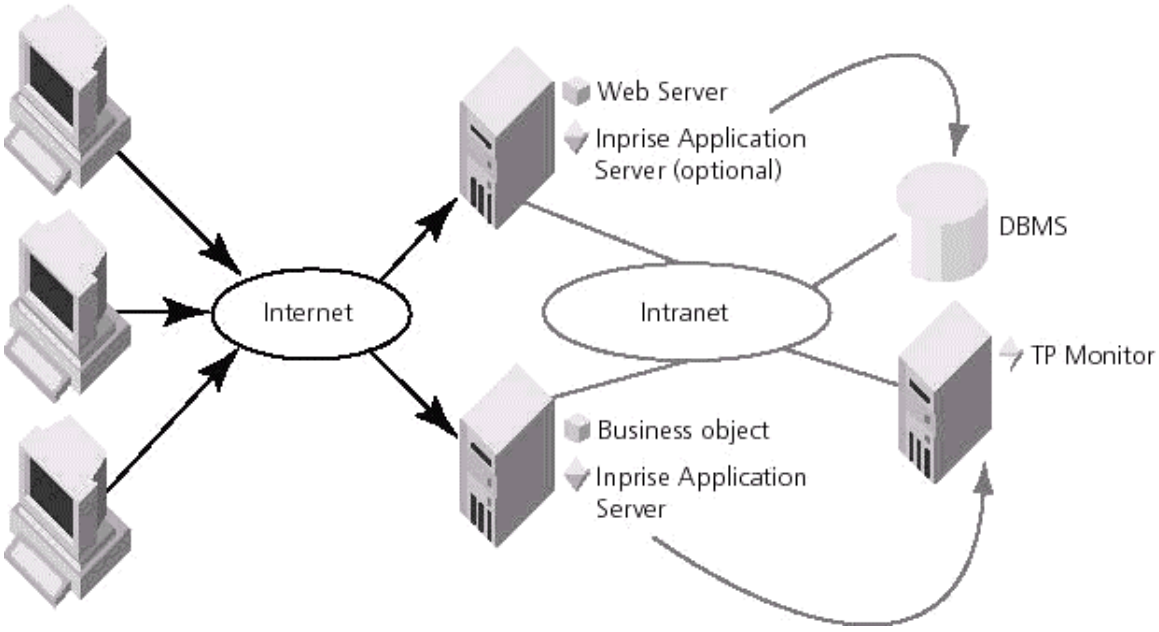

Figure 4. Borland AppServer architecture.

As shown by Figure 5, Borland AppServer handles everything necessary for enterprise applications.

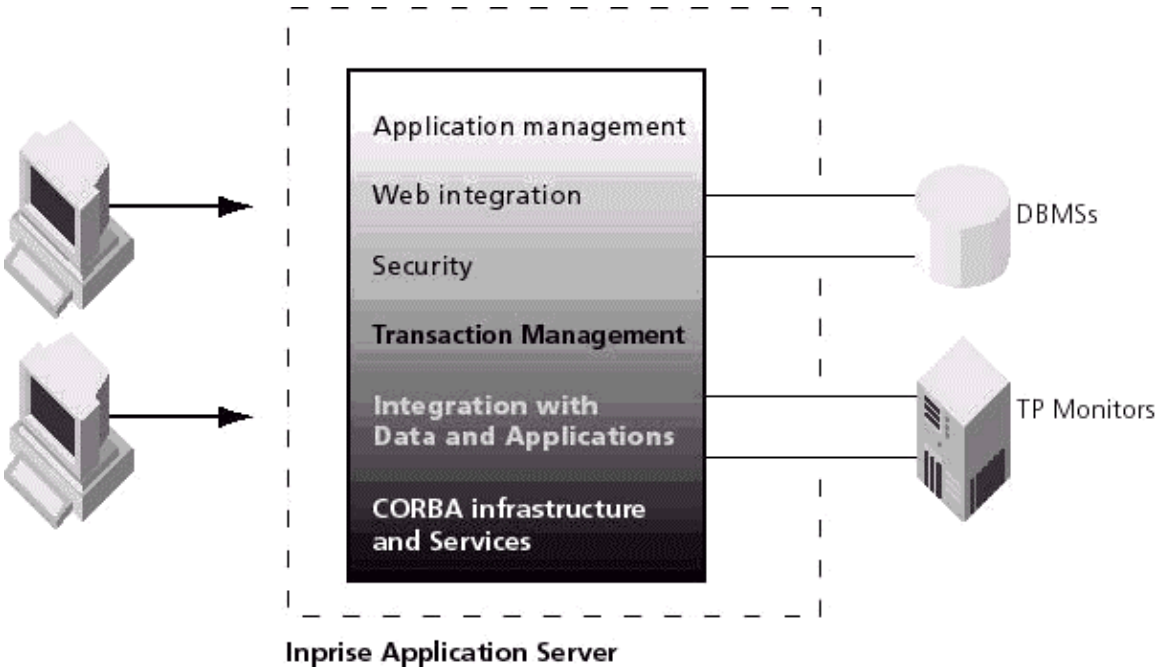

Figure 5. Functionality that meets enterprise demands—provided by Borland AppServer.

Visual Tools for Developing, Deploying and Managing

On the front end, visual tools are provided:

- **JBuilder**. This graphical development tool-the leading IDE for Java-provides a familiar visual environment for developing sophisticated distributed applications using point-and-click. It enables extensions to these applications using wizards that walk users through steps to add security and transactions to their distributed applications. Delphi and C++ Builder integration will provide complete flexibility in the choice of development tools.

- **AppCenter**. Provides a testing and deployment environment including configuration capabilities for various aspects of a distributed application. Offers centralized visual control of distributed applications, enabling administrators to set automated error detection and correction policies that ensure continued operation of mission-critical enterprise applications.

Industrial-Strength Infrastructure for Enterprise Applications

Under the covers, Borland AppServer uses core technologies:

- **VisiBroker.** This award-winning Object Request Broker (ORB) technology—for Java and C++—provides enterprise-level scalability, high availability, connection and thread management, and performance. VisiBroker for Java—the first Java implementation of the CORBA specification—includes the Gatekeeper, an extension that manages object communication from a Web server.

- **VisiBroker Integrated Transaction Service (ITS).** VisiBroker ITS handles transactions for distributed applications, ensuring atomicity, consistency, isolation, and durability for all transactions. VisiBroker ITS is compliant with the CORBA Object Transaction Service (OTS) specification, and is fully compliant with the Java Transaction Service (JTS) specification. VisiBroker ITS integrates with popular databases and mainframes, and provides seamless access to multiple data sources, supporting both XA and non-XA environments. Data from existing systems (such as TP Monitors and messaging software) can be accessed via VisiBroker ITS as well, providing true connectivity across the enterprise.

- **VisiBroker SSL (Secure Sockets Layer).** VisiBroker SSL provides a security solution built from the ground up to fully utilize the Internet and Web while providing a secure, reliable approach to the execution of transactions in a distributed environment. Its security model is based on public key cryptography (X509 certificates) and Secure Socket Layer (SSL) over IIOP. This security model ensures authentication and encryption while providing enterprise-level scalability.

- **VisiBroker Naming and Events Services.** These services are based on CORBAservices specifications from the Object Management Group (OMG), and offer simplified management of objects and events for distributed applications. Borland —the leader in Java-based CORBA technology—provided the first Java implementation of these services.

- **Web Server.** Borland AppServer bundles a Web server to enable Web deployment for distributed, multi-tier applications.

With a native implementation of IIOP—the industry communication standard for the Internet and intranets—Borland AppServer is interoperable with Java and non-Java applications

(including C, C++, Smalltalk, Delphi, and Visual Basic applications). The Borland implementation of IIOP is the de-facto industry standard used by leaders such as Oracle, Netscape, and Silicon Graphics.

Benefits Provided By Borland AppServer

The integrated architecture of Borland AppServer enables enterprise IT to respond to business needs faster with more durable solutions. These solutions are easily developed, managed, and extended using modern technology such as Java, CORBA, and SSL.

Specifically, the Borland AppServer offers these benefits:

- Simplified development of sophisticated solutions
- Open deployments in a shared environment
- Centralized management of distributed applications
- Connectivity from a variety of clients
- Transactions in a distributed environment
- Security for enterprise data
- Enterprise-level scalability, performance, and high availability
- Standards-based for interoperability

Simplified Development of Sophisticated Solutions

Using Borland JBuilder—the leading IDE for Java—IT developers can create sophisticated distributed applications using point-and-click, drag-and-drop, and wizards. Developers can easily extend these multi-tier applications for transactions and security, and integrate multiple heterogeneous data sources—all within the familiar JBuilder environment.

Power developers who want to plunge into the depths of CORBA programming can do so using the VisiBroker ORB, VisiBroker ITS, and VisiBroker SSL application programming interfaces (APIs). By enabling both GUI and code development, Borland AppServer offers a flexible solution to meet the needs of all developers.

Variety of Clients

Using JBuilder, developers can create a variety of clients, including:

- HTML clients
- Dynamic HTML clients
- Java applets running in a Web browser
- Stand-alone Java applications

Flexible Middle-Tier Servers

Middle-tier servers built with JBuilder can interact with non-Java clients (such as C++, Visual Basic, and Delphi clients) that are built using other tools. All applications running in the Borland AppServer automatically support these client technologies. This enables developers to write server-based business logic independent of the clients that will access it, and then create the clients most appropriate for the application (for example, HTML for Internet clients, Java or C++ for intranet clients, and COM for clients using popular desktop applications).

Simplified Server-side Development

For the server side, JBuilder developers can develop Enterprise JavaBeans. Using BeansExpress, Session and Entity Beans can be created, and essential tasks can be automated (such as the creation of the Home and Remote interfaces). Server-side Java applications also can be easily developed to deliver dynamic Web content using the Servlet Wizard. Servlets can be thought of as server-side versions of applets that extend the functionality of Web servers in the same way that CGI scripts do. However, servlets offer a substantial performance improvements over CGI and are truly portable across platforms. The JBuilder Servlet Wizard handles servlets that process the output of an HTML form, as well as servlets that generate HTML to be embedded in the HTML file that specified it.

Open Deployments in a Shared Environment

Using Application Server, developers and administrators use the same tool to test and deploy distributed applications. Within this console, developers can deploy distributed applications into test environments, ensure they are operating correctly, and then configure attributes and network deployments for distributed applications that will ensure the highest performance. Administrators can deploy these optimized distributed applications into production environments.

Application Server provides a centralized view of which components reside on each machine in the network, simplifying administration of the enterprise's distributed applications.

Managing components of distributed applications is also made easier with the built-in view of the Naming Service, Interface, and Implementation repositories. Administrators can use Application Server to manage their objects—they can rename objects, modify object hierarchies, and create and register new objects. Performance can also be monitored, and steps can be taken to correct problems. Application Server is cross-platform, supporting the diverse environments of the enterprise.

Centralized Management of Distributed Applications

AppCenter offers centralized management of decentralized, distributed applications. Using AppCenter, administrators can view all components of their distributed applications across the network, and set properties to configure the behavior of these applications.

Developers can perform final testing on distributed applications within AppCenter—such as tuning for fault tolerance, load balancing, and scalability—and pass these configuration policies directly on to administrators.

Automated Problem Detection and Correction

To ensure the continued success of these distributed applications, AppCenter works to solve problems before they affect users. This is achieved using automated, proactive management. AppCenter continually monitors the status of all applications under its control. If any component fails, AppCenter can be configured to take automated recovery actions to restore functionality.

Administrators determine whether AppCenter will ensure recovery by restarting failed components, or by working with the VisiBroker ORB to redirect client traffic to other instances of those components on the network.

Performance Monitoring

Administrators have a view of performance bottlenecks from within AppCenter, as well as the capability to correct problems as they arise. Although the applications monitored by AppCenter are distributed across the enterprise (and potentially across the Internet), AppCenter centralizes management of these applications to simplify administration.

Centralized Application Configuration

AppCenter allows administrators to configure several operational characteristics for their distributed applications, including:

- **Status propagation.** Administrators can set rules to help AppCenter determine whether an application is functioning correctly. For example, if two copies of an object perform the same function, an administrator can set a rule that only one of these objects needs to be active in order for the application to be functioning correctly.
- **Dependencies between components.** The order in which application objects should be started can be specified—this is helpful when a distributed application has dependencies between its components that must be maintained. Other characteristics can be set as

well; for example, administrators can specify the behavior taken by dependent objects when the object they depend upon fails.

- **Load balancing and grouping.** Objects that provide the same functionality can be grouped together, and then attributes can be set for the group. Group attributes can determine such things as the number of object instances that must be started to maintain the desired level of performance.
- **Backup or standby relationships.** By designating certain objects to be backups, administrators can maintain fault tolerance without impacting system resources. Backup objects are started only if the primary object fails.
- **Hosts relationships.** A graphical representation of host relationships shows administrators which objects reside on which computers. This aids with situations where administrators must correct system problems to restore application functionality.

Configuration information for all of the objects in an application can also be maintained using AppCenter. This information is stored in an accessible central repository. Configuration information includes various runtime parameters of the managed objects (such as command line parameters), contents of initialization files, and environment variables.

Customizable Management

And because different users have different management requirements, AppCenter supports a number of different user interface modes—ranging from a single icon that represents the status of the entire organization, to a sophisticated configuration and tuning environment. In addition, AppCenter provides a cockpit mode where customized views of the IT environment can be constructed, and then saved for later re-use.

Connectivity from a Variety of Clients

Middle-tier server components are only useful if clients can access them to request service. Borland AppServer provides connectivity for all these client varieties:

- HTML clients
- Dynamic HTML clients to handle anonymous browser-based users
- Applets running in a browser
- Stand-alone Java clients where the Java application might be used from the intranet
- Non-Java clients (such as C++ clients accessing EJB server-based components)
- Clients developed for other frameworks (like COM)

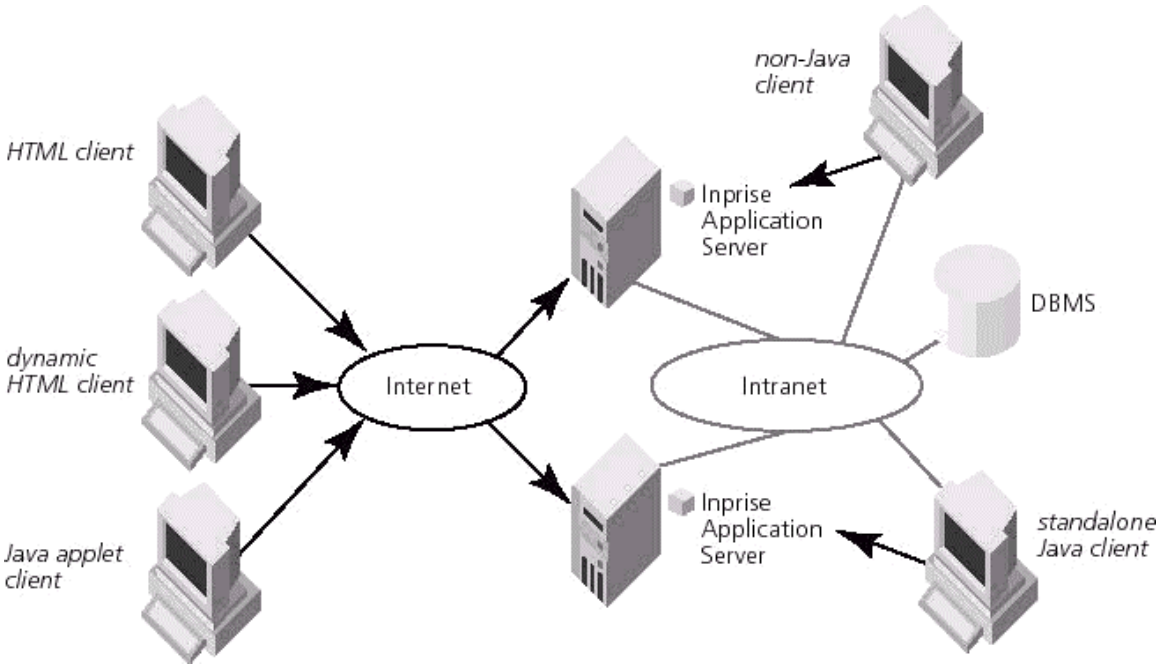

Figure 6. Connectivity with a variety of clients.

Transactions in a Distributed Environment

Through its VisiBroker ITS component, Borland AppServer coordinates two-phase commit for components of distributed applications, and ensures atomicity, consistency, isolation, and durability for all transactions.

If there is only one Resource (database or other) involved in a transaction, Borland AppServer performs a one-phase commit protocol. This means that Borland AppServer does not need to log to the disk, thereby improving performance.

Furthermore, through the VisiBroker ITS component, Borland AppServer provides connectivity to mainframes and popular databases from EJB and CORBA applications. Borland AppServer provides seamless access to multiple data sources, supporting both XA and non-XA environments. Data from existing systems (such as TP Monitors and messaging software) can be accessed as well, providing true connectivity across the enterprise.

Figure 7 shows how ITS coordinates a transaction that involves two Resources—inventory and accounts—which reside in two different types of systems (database and TP monitor).

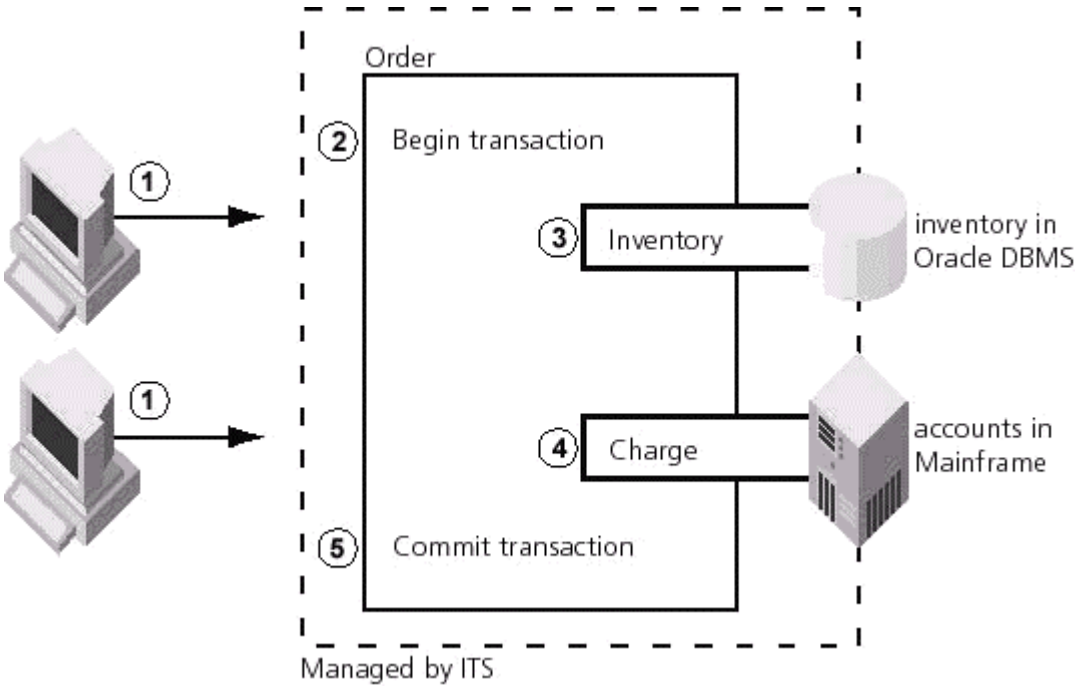

Figure 7. Robust transaction management with ITS.

Security for Enterprise Data

Enterprises require a security solution that provides privacy, integrity, and authentication. Borland AppServer provides secure communication between components of a distributed application, protects against accidental or malicious corruption, encrypts data for privacy, and authenticates clients and servers.

Additionally, this security—provided by the VisiBroker SSL component—is built from the ground up to fully utilize the Internet and Web while providing a secure, reliable approach to the execution of transactions in a distributed environment. Its security model is based on public key cryptography (X509 certificates) and Secure Socket Layer (SSL) over IIOP. This security model ensures authentication and encryption while providing enterprise-level scalability.

Enterprise-Level Scalability, Performance, and High Availability

The Borland AppServer provides enterprise-level scalability, performance, and high availability using several features of its underlying technologies:

- **Thread and connection management** is provided by the VisiBroker ORB to ensure high performance and scalability for multi-tier, distributed applications. A choice of thread management policies is available—including thread pooling for sharing server-side threads across applications—to provide flexibility for different enterprise environments. VisiBroker's connection management improves performance and scalability by multiplexing connections.
- **Fault tolerance** for distributed applications is ensured by the VisiBroker Smart Agent, a component of the VisiBroker ORB. The Smart Agent provides a configuration-free, self-healing system that can dynamically balance the load between replicas of server-based components. Using replication and distribution, the Borland AppServer eliminates any bottlenecks or single points of failure, and supports high availability to enable reliable access to critical business operations.
- **Database connection management** is handled by the VisiBroker ITS component for improved performance. By multiplexing connections to databases rather than opening new database connections for each request, the Borland AppServer saves system resources and improves scalability.

Standards-Based For Interoperability

Borland AppServer is based on industry-standard languages and protocols to ensure the highest flexibility for enterprises. Fully compliant with the CORBA specification from the Object Management Group (OMG), Borland products use the industry-standard IIOP protocol (proposed by Borland) for all object communication.

Borland AppServer also supports the XA protocol for database access, and Secure Sockets Layer (SSL) for secure Internet and intranet communication. Many Borland AppServer components are implemented in Java, and Borland AppServer uses leading Java technology: the VisiBroker for Java ORB was the first Java implementation of the CORBA ORB, VisiBroker ITS was the first Java implementation of the CORBA Transaction Service, and the VisiBroker Naming and Events services were the first Java implementations of the CORBA Naming and Events Services.

Summary

The transition to multi-tier application servers is the necessary next step in the evolution of distributed object computing. As this transition progresses, Borland has a mission to provide an integrated environment that simplifies the development, deployment, and management of multi-tier, distributed applications.

With Borland AppServer, Borland offers a powerful solution to the problem of distributed, multi-tier applications. Borland AppServer provides everything the enterprise needs: from industry-standard protocols, to integration with popular IDEs, to a robust framework for secure two-phase commit across heterogeneous data sources.

By reducing the development of distributed, middle-tier server objects down to point-and-click operations, the Borland AppServer enables the assembly of applications from reusable business logic components that reside on the middle tier. As a result, mainstream IT will be able to develop (and later, make modifications to) sophisticated enterprise applications in record time.

And by providing deployment and management tools built from the ground up to handle the rigors of distributed applications, Borland AppServer gives enterprises essential administration tools. Using these tools, administrators will be able to deploy distributed applications easily, and then manage them from a centralized location.

Ultimately, the enterprise will be able to respond quickly to market changes, and compete in the growing Web marketplace.

About Borland

Borland Corporation is playing a key role in creating the foundation on which mission-critical applications of tomorrow will be written-the open, distributed, object-based architecture for the new global enterprise. Building on its leadership and expertise in standards-based distributed object and data access technologies, Borland is a pioneer in managing distributed business logic and its access to data.

Borland distributed object products include VisiBroker for Java and VisiBroker for C++, VisiBroker ITS, VisiBroker SSL, VisiBroker Naming and Events Services, AppCenter, Delphi, C++Builder, and JBuilder.